IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

In re Application of:

Applicant:	Mortazavi et al.		
App. No.:	09/865,978	Con. No.:	6345
Filed:	May 25, 2001	Art Unit:	2152
Title:	METHOD AND APPARATUS FOR ASYNCHRONOUS COMPONENT INVOCATION	Examiner:	Lesniewski, Victor D.

APPEAL BRIEF IN SUPPORT OF APPELLANT'S APPEAL TO THE BOARD OF PATENT APPEALS AND INTERFERENCES

MAIL STOP APPEAL BRIEF - PATENTS
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Sir:

Applicant (hereafter "Appellant") hereby submits this Brief in support of its appeal from a final decision by the Examiner, mailed August 22, 2006 in the above-captioned case. Appellant respectfully requests consideration of this appeal by the Board of Patent Appeals and Interferences (hereafter the "Board") for allowance of the above-captioned patent application.

An oral hearing is not requested at this time.

TABLE OF CONTENTS

I. REAL PARTY IN INTEREST

The invention is assigned to Sun Microsystems, Inc. of 4150 Network Circle, Santa Clara, California 95054.

II. RELATED APPEALS AND INTERFERENCES

To the best of Appellant's knowledge, there are no appeals or interferences that are related to, will directly affect, will be directly affected by, or have a bearing on the Board's decision in the present appeal.

III. STATUS OF THE CLAIMS

Claims 1, 4-11, 13-16, 19-26, 28-31, 35-38 and 42-43 are currently pending in the above-referenced application. No claims have been allowed.

IV. STATUS OF AMENDMENTS

Claims 1, 4-11, 13-33, 35-40 and 42-43 were finally rejected under 35 U.S.C. § 102(e) in the final Office action mailed August 22, 2006. In response to the final Office action mailed August 22, 2006, Appellant filed a response after final pursuant to 37 C.F.R. § 1.116 on September 15, 2006, canceling various claims with claims 1, 4-11, 13-16, 19-26, 28-31, 35-38 and 42-43 remaining pending. Subsequently, an Advisory action was mailed on September 28, 2006 entering the amendment after final and maintaining all rejections under 35 U.S.C. § 102(e) from the final Office action. A copy of all claims on appeal is attached hereto as the Appendix of Claims.

Appellant respectfully traverse each of the grounds of rejection.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

Independent claims 1, 16 and 43 are similar in inventive scope. Independent claim 1 discloses a computer implemented method, independent claim 16 discloses a computer program product, and independent claim 43 discloses an apparatus set forth in means plus function form pursuant to 35 U.S.C. § 112, sixth paragraph. As such, claims 1, 16 and 43 disclose the operations of a first component asynchronously invoking a second component in an object-oriented computing environment. *See Specification page 5, lines 4-7.* The operations include receiving at an asynchronous proxy an asynchronous request from a first object-oriented component that resides at a second server. *See Specification page 5, lines 7-8.* The request has a void return type and is not associated with application-specific exceptions, which allows for asynchronous invocations while maintaining an efficient programming model. *See Specification page 16, lines 2-7.* Further, an exception listener is set on the asynchronous proxy as well as a scope of the second component. An exception listener handles exceptions associated with the asynchronous invocations, and the exception listener is registered for the second component. *See Specification page 5, lines 29-30.* Furthermore, the request and the scope are stored in a queue on the asynchronous proxy, and a thread for identifying the received request and invoking the second component is provided. *See Specification page 11, line 31 – page 12, line 2.* The thread identifies an exception listener object-oriented component for handling exceptions associated with the invocation of the second component. *See Specification page 5, lines 8-11.* The exception listener registered on an asynchronous proxy, is stateless. Stateless means that the exception listener is operable to handle a plurality of types of exceptions from a plurality of different components. *See Specification page 12, lines 4-11.* Due to the fact that the exception listener is stateless a client does not expect a return from asynchronous proxy, and can therefore proceed unhindered with transaction processing. *See Specification page 12, lines 7-8.*

Independent claims 10, 25, 31 and 38 are similar in inventive scope. Independent claim 10 discloses a computer-implemented method, independent claims 25 and 31 disclose a computer program product, and independent claim 38 discloses a computer system. As such, claims 10, 25, 31 and 38 disclose the operations of a first object-oriented component asynchronously invoking a second object-oriented component in an object-oriented environment. *See Specification page 5, lines 4-7.* The operations include transmitting an asynchronous request from the first object-oriented component residing at a first server in order to invoke the second object-oriented component residing at a second server. *See Specification page 5, lines 7-8.* The first and second object-oriented components operate in environments that allow direct invocation of the second component by the first component.

Further, the operations include receiving the asynchronous request which has a void return type and is not associated with application-specific exceptions. *See Specification page 16, lines 4-5.* An exception listener object-oriented component on an asynchronous proxy and a scope of the second object-oriented components is registered. *See Specification page 11, line 31 – page 12, line 2.* The exception listener being registered for the second component, and the asynchronous proxy being associated with the second component. The exception listener is stateless, and is operable to handle a plurality of types of exceptions from a plurality of different components. *See Specification page 12, lines 4-11.*

VI. GROUNDS OF REJECTION

A. Claims 1, 4-11, 13-16, 19-26, 28-31, 35-38 and 42-43 are rejected under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent Number 6,804,818 to Codella, et al. (hereafter "Codella").

No claims were indicated as allowable.

VII. ARGUMENT

THE REJECTION OF CLAIMS 1, 4-11, 13-16, 19-26, 28-31, 35-38 AND 42-43 UNDER 35 U.S.C. § 102(e) IS IMPROPER BECAUSE THE REFERENCE DOES NOT SHOW ALL THE ELEMENTS OF THE CLAIMS

The Examiner has rejected claims 1, 4-11, 13-16, 19-26, 28-31, 35-38 and 42-43 under 35 U.S.C. § 102(e) as being anticipated by Codella. In order for a claim to be anticipated under 35 U.S.C. § 102, "each and every element as set forth in the claim is found, either expressly or inherently described, in a single prior art reference." See MPEP § 2131, See *also Verdegaal Bros. v. Union Oil Co. of California*, 841 F.2d 628, 631, 2 USPQ2d 1051, 1053 (Fed. Cir. 1987). For the reasons recited below, it is respectfully submitted that Codella does not anticipate the claims included in the Appendix of Claims.

A. Independent claims 1, 10, 16, 25, 31, 38 and 43 are patentable over Codella

Claims 1, 10, 16, 25, 31, 38 and 43 are independent claims from which all other pending claims depend. Claims 10, 16, 25, 31, 38 and 43 include nearly the same limitations and are rejected on the same grounds as claim 1. Accordingly, our initial arguments will focus on the independent claims.

1. Codella fails to disclose that the asynchronous request has a void type return and is not associated with application-specific exceptions

Appellant submits that Codella does not disclose that "the asynchronous request has a void type return and is not associated with application-specific exceptions" as recited by claim 1, and similarly recited by claims 10, 16, 25, 31, 38 and 43. (emphasis provided). The Examiner asserts that Codella at column 9, lines 31-33 and column 10, lines 61-62 discloses this feature of claim 1. *See final Office action, page 4, section 17.* However, column 9, lines 31-33 of Codella discloses that a "message proxy 104... has a no return type" and column 10, lines 61-62 discloses that "message formats may include conventional, application-independent field-names." Even though Codella discloses no return type, Appellant submits that applying a no-return type to a message proxy as opposed to an asynchronous request is completely different. Stated differently, because an asynchronous request and a message proxy have completely different functionality, simply having a similar returns type does not render them the same. Furthermore, Codella discloses an application independent field name, field name not an application specific exception.

Specifically, a message proxy "abstracts [a] target of [an] invocation by associating its methods with target destinations." *See Codella*, col. 9, lines 15-17. While an asynchronous request is a request that allows a "client [to] continue processing or receiving messages while [an] invocation of [a] remote component occurs." *See Specification, page*

16, lines 2-4. Accordingly, an asynchronous request and a message proxy perform completely different functions. A message proxy associates methods with target destinations while an asynchronous request allows continued processing while remote invocation of components occur. Accordingly, a message proxy is not the same as an asynchronous request.

Additionally, claim 1's application-specific exception is not the same as Codella's application-independent field name. First, the Cambridge dictionary defines "specific" as "relating to one thing and not others (i.e. particular)", while it defines "independent" as "not influenced or controlled in any way." As such, the meanings of specific and independent directly contradict each other; therefore, an application-independent field name cannot be the same as an application-specific exception. Second, Appellant submits that an exception and a field name are not the same. In object-oriented programming, an exception is related to an error encountered while executing a program, whereas a field name is a name assigned to a variable passed to an object or method when invoking the object or method. Accordingly, a field name is not an exception.

Thus, for any one of the above recited reasons, independent claims 1, 10, 16, 25, 31, 38 and 43 are not anticipated under 35 U.S.C. § 102(e) by Codella.

2. Codella fails to disclose that the exception listener is stateless

Appellant submits that Codella does not disclose that "the exception listener… is stateless" as recited by claim 1, and similarly recited by claims 10, 16, 25, 31, 38 and 43. (emphasis provided). The Examiner asserts that a "stateless session bean [of which] the message bean listener only needs to maintain a single instance of" (*See Codella* column 17, lines 22-24) is the same as the stateless exception listener of claim 1. *See final Office action, page 4, section 17*. Appellant respectfully disagrees with the Examiner's assertion. Appellant respectfully submits that a reference merely including the same word as in a claim (i.e. stateless) is insufficient to show anticipation under 35 U.S.C. § 102(e). Specifically, MPEP § 2131 states that "[t]he identical invention must be shown in as complete detail as is contained in the… claim." *See also Richardson v. Suzuki Motor Co.*, 868 F.2d 1226 (Fed. Cir. 1989) (emphasis provided). In this case, Codella simply includes the word "stateless" and is lacking the necessary details to show the identical invention of claim 1. Stated differently, an exception listener is not the same as a session bean, irrespective of whether both are stateless.

Specifically, Codella defines session beans as beans that are "inherently non-shareable by virtue of being session beans, [but] can be effectively shared across clients given that they only supply behavior and not state data." *See Codella, col. 10, lines 33-36*. Whereas, the Specification at page 5, lines 28-30 defines an exception listener as a program

that "uses a scope corresponding to [a] request to handle exceptions associated with [an] invocation." As such, a session bean is clearly not the same as an exception listener, even though Codella's session bean and claim 1's exception listener are both stateless.

Thus, for at least these additional reasons, independent claims 1, 10, 16, 25, 31, 38 and 43 are not anticipated under 35 U.S.C. § 102(e) by Codella.

B. Dependent claims are not anticipated by Codella

Dependent claims 4-9, 11, 13-15, 19-24, 26, 28-30, 35-37 and 42 depend upon and contain all the limitations of independent claims 1, 10, 16, 25, 31 and 38, respectively. Therefore, for at least the reasons mentioned above, Codella fails to disclose each and every limitation of claims 44-9, 11, 13-15, 19-24, 26, 28-30, 35-37 and 42. As such, claims 4-9, 11, 13-15, 19-24, 26, 28-30, 35-37 and 42 are patentable under 35 U.S.C. § 102(e) over Codella.

VIII. CONCLUSION

Appellant respectfully submits that all the appealed claims in this application are patentable and requests that the Board of Patent Appeals and Interferences direct allowance of the rejected claims.

The Appellant believes that no fees or petitions are required are required at this time. However, if any such petitions or fees are necessary, please consider this a request therefor and authorization to charge Deposit Account No. 04-1415 accordingly.

Dated: April 2, 2007

Respectfully submitted,

Gregory P. Durbin, Registration No. 42,503
Attorney for Appellant
USPTO Customer No. 66083

DORSEY & WHITNEY LLP
Republic Plaza Building, Suite 4700
370 Seventeenth Street
Denver, Colorado 80202-5647
Phone: (303) 629-3400
Fax: (303) 629-3450

IX. APPENDIX OF CLAIMS

1. A computer-implemented method for a first component to invoke a second component asynchronously in an object-oriented computing environment, the computer-implemented method comprising:

receiving at an asynchronous proxy an asynchronous request from a first object-oriented component residing at a second server wherein the request has a void return type and is not associated with application-specific exceptions;

setting an exception listener on the asynchronous proxy and a scope of the second component, the exception listener being registered for the second component;

storing the request and the scope in a queue on the asynchronous proxy; and

providing a thread for identifying the received request and invoking the second component, wherein the thread identifies an exception listener object-oriented component for handling exceptions associated with the invocation of the second component, wherein the exception listener is registered on an asynchronous proxy, is stateless and is operable to handle a plurality of types of exceptions from a plurality of different components.

4. The computer-implemented method of claim 1, wherein the first and second components reside in environments allowing components to directly invoke other components.

5. The computer-implemented method of claim 1, wherein the first and second components are Enterprise Java Bean components.

6. The computer-implemented method of claim 5, wherein the first and second components are associated with a container.

7. The computer-implemented method of claim 6, further comprising placing the request from the first component in a queue.

8. The computer-implemented method of claim 7, wherein a worker thread dequeues the received request after receiving a transaction commit signal from the container.

9. The computer-implemented method of claim 8, wherein the exception listener receives the exception and the scope of the exception.

10. A computer-implemented method for a first object-oriented component to invoke a second object-oriented component asynchronously in an object-oriented environment, the computer-implemented method comprising:

transmitting an asynchronous request from the first object-oriented component residing at a first server to invoke the second object-oriented component residing at a second server, the first and second object-oriented components operating in environments allowing direct invocation of the second component by the first component;

receiving the asynchronous request wherein the request has a void return type and is not associated with application-specific exceptions; and

registering an exception listener object-oriented component on an asynchronous proxy and a scope of the second object-oriented components, the exception listener being registered for the second component and, the asynchronous proxy being associated with the second component, wherein the exception listener is stateless and is operable to handle a plurality of types of exceptions from a plurality of different components.

11. The computer-implemented method of claim 10, wherein the asynchronous proxy has the same type as the second component.

13. The computer-implemented method of claim 10, wherein the first and second components are associated with separate servers.

14. The computer-implemented method of claim 10, wherein the first and second components are Enterprise Java Bean components.

15. The computer-implemented method of claim 14, wherein the first and second components are associated with a container.

16. A computer program product comprising computer code for a first component to invoke a second component asynchronously, the computer program product comprising:

computer code for receiving at an asynchronous proxy an asynchronous request form a first object-oriented component residing at a first server to invoke a second object-oriented component residing at a second server, wherein the request has a void return type and is not associated with application-specific exceptions;

computer code for setting an exception listener on the asynchronous proxy and a scope of the second component, the exception listener be registered for the second component;

computer code for storing the request and the scope in a queue on the asynchronous proxy;

computer code for providing a thread for identifying the received request and invoking the second component, wherein the thread identifies an exception listener object-oriented component for handling exceptions associated with the invocation of the second component, wherein the exception listener is registered on an asynchronous proxy, is stateless and is operable to handle a plurality of types of exceptions from a plurality of different components; and

a computer-readable medium for storing the computer codes.

19. The computer program product of claim 16, wherein the first and second components reside in environments allowing components to directly invoke other components, wherein the first and second components are associated with separate servers.

20. The computer program product of claim 16, wherein the first and second components are Enterprise Java Bean components.

21. The computer program product of claim 20, wherein the first and second components are associated with a container.

22. The computer program product of claim 21, further comprising placing the request from the first component is placed in a queue.

23. The computer program product of claim 22, wherein the worker thread dequeues the received request after receiving a transaction commit signal from the container.

24. The computer program product of claim 23, wherein the exception listener receives the exception and the scope of the exception.

25. A computer program product for a first object-oriented component to invoke a second object-oriented component asynchronously in an object-oriented environment, the computer program product comprising:

computer code for transmitting an asynchronous request from a first object-oriented component residing at the first server to invoke the second object-oriented component

residing at a second server, the first and second object-oriented components operating in environments allowing direct invocation of the second component by the first component;

computer code for receiving the asynchronous request wherein the request has a void return type and is not associated with application-specific exceptions;

computer code for registering an exception listener object-oriented component on an asynchronous proxy, and for setting a scope associated with the second object-oriented components, the exception listener being registered for the second component and the asynchronous proxy associated with the second component, wherein the exception listener is stateless and is operable to handle a plurality of types of exceptions from a plurality of different components; and

a computer-readable medium for storing the computer codes.

26. The computer program product of claim 25, wherein the asynchronous proxy has the same type as the second component.

28. The computer program product of claim 25, wherein the first and second components are associated with separate servers.

29. The computer program product of claim 25, wherein the first and second components are Enterprise Java Bean components.

30. The computer program product of claim 29, wherein the first and second components are associated with a container.

31. A computer program product for an enterprise environment associated with a computing system, the computer program product comprising:

an asynchronous proxy for receiving a request from a first object-oriented component residing at a first server, the request intending to invoke a second object-oriented component residing at a second server wherein the request has a void return type and is not associated with application-specific exceptions; and

an exception listener object-oriented component coupled to the asynchronous proxy and registered for the second object-oriented component, wherein the exception listener uses a scope corresponding to the request to handle exceptions associated with the invocation of the second object-oriented component, wherein the exception listener is stateless and is operable to handle a plurality of types of exceptions from a plurality of different components.

35. The computer program product of claim 31, wherein the first and second components are Enterprise Java Bean components.

36. The computer program product of claim 35, wherein the first and second components are associated with a container.

37. The computer program product of claim 31, wherein the worker thread invokes the second components after receiving a transaction commit signal from the container.

38. A computer system operating an enterprise environment, the computer system comprising:

a processor coupled to memory; and

an interface coupled to the processor, the interface configured to transmit a request from a first object-oriented component residing at the first server to invoke a second object-oriented component residing at a second server, wherein the request has a void return type and is not associated with application-specific exceptions the first and second object-oriented components operating in environments allowing direct invocation of the second component by the first component, wherein the interface also transmits information to register an exception listener on an asynchronous proxy, the asynchronous proxy associated with the second component and the exception listener being registered for the second component and, wherein the exception listener is stateless and is operable to handle a plurality of types of exceptions from a plurality of different components.

42. The computer system of claim 38, wherein the first and second components are Enterprise Java Bean components.

43. An apparatus for a first component to invoke a second component asynchronously in an object-oriented computing environment, the apparatus comprising:

means for receiving at an asynchronous proxy an asynchronous request from a first object-oriented component residing at a first server to invoke a second object-oriented component residing at a second server, wherein the request has a void return type and is not associated with application-specific exceptions;

means for setting an exception listener on the asynchronous proxy and a scope of the second component, the exception listener being registered for the second component;

means for storing the request and the scope in a queue on the asynchronous proxy; and

means for providing a thread for identifying the received request and invoking the second component, wherein the thread identifies an exception listener object-oriented component for handling exceptions associated with the invocation of the second component, the exception listener object-oriented component registered on an asynchronous proxy, wherein the request is associated with no application specific exceptions.

X. EVIDENCE APPENDIX

None.

XI. RELATED PROCEEDINGS APPENDIX

None.